Abstract p-time proof nets for MALL: Conflict nets

Dominic J. D. Hughes*

Stanford University

January 11, 2007

This paper presents proof nets for multiplicative-additive linear logic (MALL), called *conflict nets*. They are *efficient*, since both correctness and translation from a proof are p-time (polynomial time), and *abstract*, since they are invariant under transposing adjacent &-rules.

A conflict net on a sequent is concise: axiom links with a conflict relation. Conflict nets are a variant of (and were inspired by) *combinatorial proofs* introduced recently for classical logic: each can be viewed as a maximal map (homomorphism) of contractible coherence spaces (P_4-free graphs, or cographs), from axioms to sequent.

The paper presents new results for other proof nets: (1) correctness and cut elimination for slice nets (Hughes / van Glabbeek 2003) are p-time, and (2) the cut elimination proposed for monomial nets (Girard 1996) does not work. The subtleties which break monomial net cut elimination also apply to conflict nets: as with monomial nets, existence of a confluent cut elimination remains an open question.

1 Introduction

Jean-Yves Girard's seminal paper [Gir87] on linear logic introduced an elegant abstract representation of a proof called a *proof net*. These original proof nets used *boxes* [Gir87, p. 45] to deal with the superposition associated with &-connectives. Boxes mimic the sequent calculus &-rule almost directly, so that the following two proofs, which differ only in the order of adjacent &-rules, have distinct box nets:

$$\dfrac{\dfrac{\dfrac{}{\mathsf{P},\overline{\mathsf{P}}}\quad\dfrac{}{\mathsf{P},\overline{\mathsf{P}}}}{\mathsf{P}\,\&\,\mathsf{P},\overline{\mathsf{P}}}\,\&\qquad\dfrac{\dfrac{}{\mathsf{P},\overline{\mathsf{P}}}\quad\dfrac{}{\mathsf{P},\overline{\mathsf{P}}}}{\mathsf{P}\,\&\,\mathsf{P},\overline{\mathsf{P}}}\,\&}{\mathsf{P}\,\&\,\mathsf{P},\overline{\mathsf{P}}\,\&'\,\overline{\mathsf{P}}}\,\&' \qquad\qquad \dfrac{\dfrac{\dfrac{}{\mathsf{P},\overline{\mathsf{P}}}\quad\dfrac{}{\mathsf{P},\overline{\mathsf{P}}}}{\mathsf{P},\overline{\mathsf{P}}\,\&'\,\overline{\mathsf{P}}}\,\&'\qquad\dfrac{\dfrac{}{\mathsf{P},\overline{\mathsf{P}}}\quad\dfrac{}{\mathsf{P},\overline{\mathsf{P}}}}{\mathsf{P},\overline{\mathsf{P}}\,\&'\,\overline{\mathsf{P}}}\,\&'}{\mathsf{P}\,\&\,\mathsf{P},\overline{\mathsf{P}}\,\&'\,\overline{\mathsf{P}}}\,\&$$

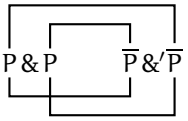
(The marked connective $\&'$ is for distinction, we omit sequent turnstiles $\vdash$, and $\overline{\mathsf{P}}$ is the dual of P.)

The follow-up paper [Gir96] tried a different approach to superposition. Every & is given an *eigenvariable*, and every node in the proof net has a list of possibly-negated eigenvariables, its *monomial*. Monomial nets suffer two main defects relative to box nets:

*Visiting Scholar, Computer Science Department, Stanford University, CA 94305.

- There is no canonical surjection from cut-free proofs to monomial nets.[1] One can no longer ask "Which proofs are identified upon translation to a proof net?": monomial nets fail to provide a semantics for cut-free proofs.[2]
- Unfortunately the cut elimination proposed for monomial nets [Gir87, p. 24] does not work: Section 10 gives a counterexample. Existence of a confluent cut elimination remains an open question.[3]

The *slice nets* of [HG03, HG05] solve these problems by taking a proof net to be a set of axiom linkings, or *slices*.[4] (Equivalently, a slice net can be represented as a set of boolean-weighted axiom links.) There is a canonical surjection from proofs. For example, the two proofs above map to the following slice net, comprising four axiom linkings, each linking containing just one axiom link:[5]



Slice nets were shown to have a simple confluent cut elimination, and a hyper-elimination which occurs independently slice-by-slice (by GoI-style path composition), yielding a category [HG03, HG05]. The present paper (Section 9) proves that correctness of slice nets is p-time.

But all is not rosy with slice nets: there can be an exponential blowup in size when translating a proof.[6] [7] This is a flaw if we take seriously the notion that a semantics is a *structure preserving map*, or some kind of *homomorphism* from proofs: we are failing to respect computational complexity. A key insight of propositional proof complexity [CR79] is that complexity is important in decidable logics such as MALL.[8]

[1] There is a canonical *non*-surjective function: identify no formulas during translation [Gir96, p. 7]. The image of this function is precisely the box proof nets, disguised in monomial form. So as a semantics of cut-free proofs, this is exactly the box net semantics. Since every box proof net is a monomial proof net, there are actually *more* monomial proof nets than box proof nets.

[2] See [HG03, HG05] for a detailed explanation, with examples.

[3] One always has a trivial non-confluent cut elimination via sequentialization, which is uninteresting.

[4] This underlying data structure is mentioned in appendix A.1.6 of [Gir96]. The essential contribution of [HG03, HG05] was to provide the elusive geometric correctness criterion and exhibit a simple confluent cut elimination.

[5] Note that this is not a single linking with four axiom links; it is four linkings each with a single axiom link. In this particular case, there is also a canonical monomial net, but that is not true in general.

[6] Consider the unique cut-free proof of $\otimes^n(1\&1)$, where $\otimes^n$ denotes iterated tensor $\otimes$ with n arguments associated to the left (*e.g.* $\otimes^3 A = (A\otimes A)\otimes A$), in which $\otimes$-rules are below &-rules. Since there are n &-rules, translating this proof Π_n to a slice net θ_n blows up exponentially: θ_n has 2^n slices (axiom linkings). (For an example without the tensor unit 1, read each 1 as $a\parr\overline{a}$.) The exponential blowup when mapping to a set of slices is mentioned in Appendix A.1.6 of [Gir96].

[7] As remarked earlier, a set of slices can just as well be represented as a set of weighted axiom links (arbitrary non-monomial boolean weights, *e.g.* $p \cup q$ for eigenvariables p and q). This trivial change of notation does not eliminate the exponential blowup: with n &'s in the sequent, a boolean weight is a subset of the 2^n hypercube.

[8] In first-order logic, which is undecidable, the value of a proof as a certificate of theoremhood is absolutely clear. But in a decidable, propositional setting, what is the *point* of being handed a proof? To determine theoremhood, we only need the formula. The idea in propositional proof complexity is to reinstate and quantify the value of a proof certificate: if the correctness of a certificate can be checked in polynomial time in its size, and the certificate is not 'too big' relative to the formula, checking the certificate will be faster than than deciding the theoremhood of the formula. See [Urq95] for an accessible introduction to propositional proof complexity.

	Representation efficiency		*Abstraction*		*Cut elimination*	
Proof net	**P-time correctness**	**P-time translation**	**Raise ⅋/⊕/&-rule over &-rule**	**Raise ⊗-rule over &-rule**	**P-time**	**Confluent (unit-free)**
Box [Gir87]	✓	✓	✗	✗	✗?	?
Monomial [Gir96]	?	✓[a]	✗[a]	✗[a]	✗?	?[b]
Slice [HG03,05]	✓	✗	✓	✓	✓[d]	✓
Conflict	✓	✓	✓	✗[c]	✗?	?

✓=yes ✗=no ?=open question ✗?=open question, probably no

[a] With respect to the canonical (non-surjective) proof translation [Gir96, p. 7]. See footnote 1.
[b] The definition proposed in [Gir96, p. 24] does not work: see Section 10.
[c] Seemingly the price of having a p-time translation from proofs.
[d] P-time since normalisation is slicewise.

Table 1: Comparison of proof nets.

This paper presents a new notion of proof net, called a *conflict net*, such that:

(1) Checking correctness is p-time in the size of the proof net.

(2) Translation from a proof is p-time (improving on slice nets [HG03, HG05]).

(3) Translation is invariant under transposing adjacent &-rules, and raising a ⅋- or ⊕-rule over a &-rule (improving on box nets [Gir87] and monomial nets [Gir96][9]).

(4) Extracting a sequentialization is p-time.

(5) A conflict net on a sequent is concise: axiom links with a conflict relation.

(6) Proof translation is simple: axioms become axiom links, and two axiom links conflict iff they are from opposite branches above a &-rule.

Examples of conflict nets are shown in Figure 1. Figure 1 also illustrates how translation from a proof to a conflict net is invariant with respect to raising a ⊕/&/⅋-rule over a &-rule. Table 1 compares different proof nets.

[9] With respect to the canonical (non-surjective) proof translation [Gir96, p. 7]. See footnote 1.

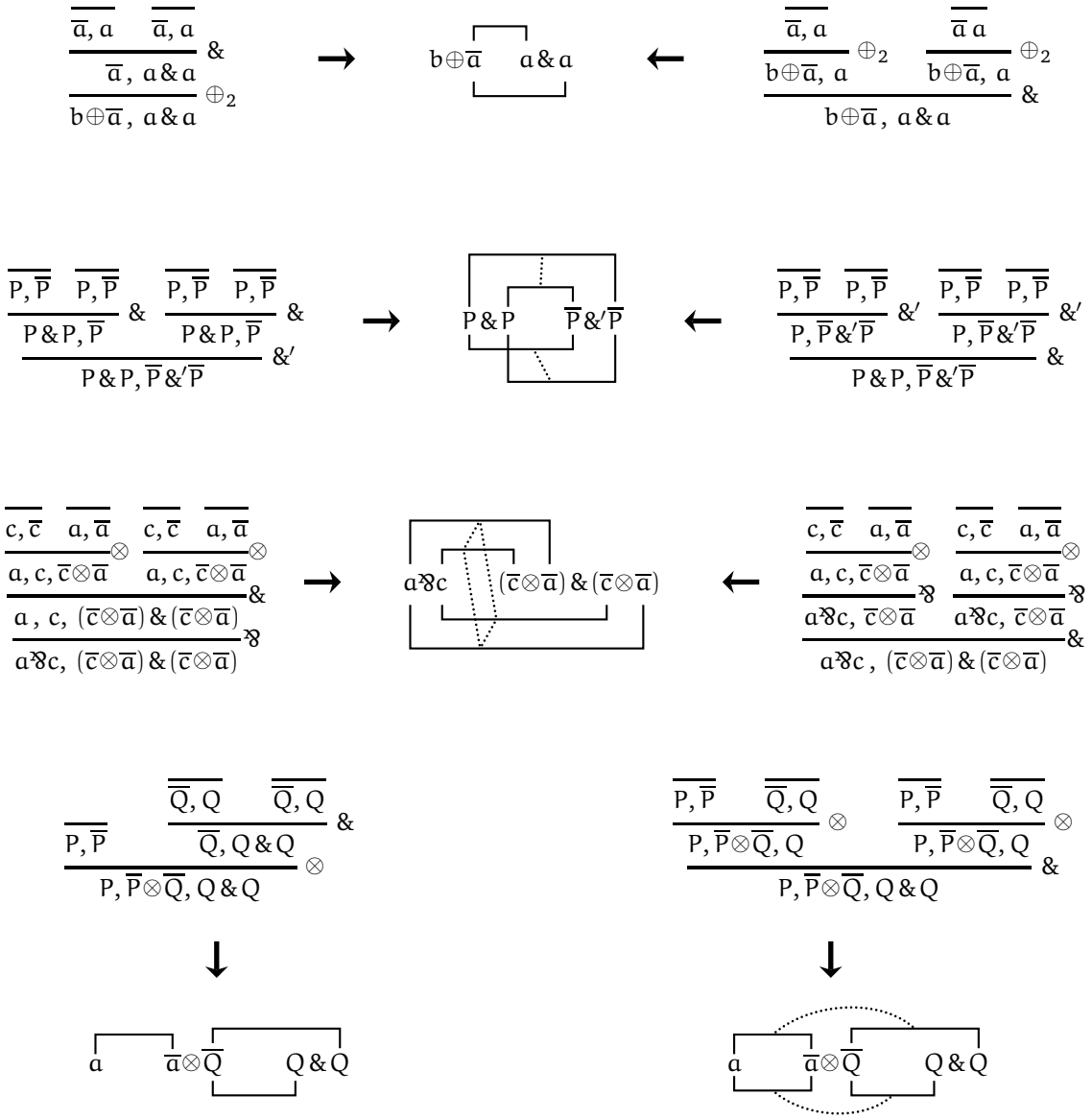


Figure 1: Illustrating the surjective translation function from proofs to conflict nets. The first three rows show how conflict nets are invariant with respect to raising a $\oplus$-, $\&$- or $\parr$-rule over a $\&$-rule, respectively: each pair of proofs (left and right) maps to the same conflict net (centre). The last two translations (flowing downwards) show that raising a $\otimes$-rule over a $\&$-rule changes the conflict net; this seems to be the price of p-time proof translation. Conflicts between axiom links are shown as dotted edges. Axiom links which overlap (share an atom in the sequent) conflict implicitly.

Related work. The last few years have seen a renaissance of work involving MALL proof nets, including [Ham04] (extending monomial nets with mix, analysing softness), [CP05] (a language for MALL proofs, viewed as processes), [CF05] (a ludics-based analysis of sequentiality/parallelism), [BHS05] (a fully complete *relational* model for MALL), [Mai07] (extending Danos contractibility [Dan90] to additives, using a distributivity rewrite), [Abr07] (a domain-theoretic view of unfolding the &-rules as we go up a proof), to name but a few.[10]

In each case the underlying data structure involved are more complex than a conflict net, carrying additional machinery such as monomial weights on subformulas, subformula occurrences, focalisation, contraction nodes, domains, partial left/right resolutions of the &'s in a sequent, and so on. Like box nets and monomial nets, most deal with occurrences of *subformulas*; the data structure of a conflict net involves only atoms, true to the spirit of the geometry of interaction [Gir89]. By not dealing with internal nodes of subformula trees, which are sequential, conflict nets are in some sense maximally parallel.

Current work for conflict nets includes arranging them into a category, possibly via a strongly normalising cut elimination. A naive cut elimination can be obtained by emulating the elimination of box nets (copying empires around). One possible approach is to try and use pullbacks of (contractible) coherence spaces to obtain a completely abstract form of cut hyper-elimination (composition) in a compact closed category. If it worked out, this would ensure a forgetful functor to the underlying compact closed composition of slice nets.

Conflict nets are a variant of (and were inspired by) *combinatorial proofs* introduced recently for classical logic [Hug06a, Hug06b]: each conflict net can be viewed as a maximal map (homomorphism) of contractible coherence spaces (P_4-free graphs, or cographs), from axioms to sequent. The relationship with combinatorial proofs is sketched in Section 11.

Acknowledgement. I'm grateful to Robin Houston for discussions about abstract categorical versions of cut elimination, based on pullbacks of coherence spaces. In particular, Robin showed me how to construct pullbacks in the category of coherence spaces. I'm also indebted to Roberto Maieli, whose extension of Danos' contractability criterion [Mai07] stimulated me to think about MALL proof nets again.

2 Preliminaries

2.1 MALL

We work with *cut-free, unit-free multiplicative-additive linear logic* [Gir87], henceforth denoted MALL.

Fix a set $\mathcal{A} = \{a, b, c, \ldots\}$ of ***literals*** equipped with a function $\overline{(\ )} : \mathcal{A} \to \mathcal{A}$ such that $\overline{a} \neq a$ and $\overline{\overline{a}} = a$ for all $a \in \mathcal{A}$. MALL formulas are generated from literals by the binary connectives $\otimes$ (tensor) $\parr$ (par) & (with) and $\oplus$ (plus). Define $\overline{\otimes} = \parr$, $\overline{\parr} = \otimes$, $\overline{\oplus} = \&$ and $\overline{\&} = \oplus$. Define negation $(.)^\perp$ by $a^\perp = \overline{a}$ on literals, and $(A \square B)^\perp = A^\perp \overline{\square} B^\perp$. Formulas A and $A^\perp$ are ***dual***. A sequent is a list (finite sequence) $A_1, \ldots, A_n$ of formulas ($n \geqslant 0$). Throughout this document we take $P, Q, R, \ldots$ to range over literals, $A, B, C, \ldots$ over formulas, and $\Gamma, \Delta, \Sigma, \ldots$ over sequents.

We identify a formula with its parse tree: a tree with leaves labelled with literals and internal vertices labelled with connectives, equipped with a linear order on leaves. Edges are oriented away

[10] With polarization, proof nets become much easier: see [LdF04].

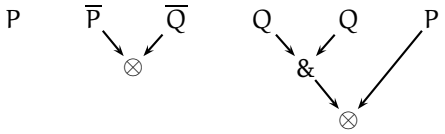
$$\frac{}{P, \overline{P}}\ \mathsf{ax} \qquad \frac{\Gamma}{\sigma\Gamma}\ \mathsf{perm}_\sigma$$

$$\frac{\Gamma,\ A\ ,\ B}{\Gamma,\ A \parr B}\ \parr \qquad \frac{\Gamma,\ \ A_i}{\Gamma,\ A_o \oplus A_1}\ \oplus_i$$

$$\frac{\Gamma, A \qquad B, \Delta}{\Gamma, A \otimes B, \Delta}\ \otimes \qquad \frac{\Gamma, A \qquad \Gamma, B}{\Gamma, A \,\&\, B}\ \&$$

Figure 2: MALL proof rules. Here σ is any permutation on n, the number of formulas in the sequent Γ above the perm-rule.

from the leaves. We identify a sequent with its parse forest: the disjoint union of its formulas (formula parse trees), with a linear order on leaves. For example, we identify the three-formula sequent $P\ ,\ \overline{P} \otimes \overline{Q}\ ,\ (Q \,\&\, Q) \otimes P$ with the following parse forest:



The linear order on leaves is given by the left-to-right order on the page. Two leaves are ***dual*** if their literal labels are dual.

If $\Gamma = A_1, \ldots, A_n$, and σ be a permutation on n (*i.e.*, a bijection $\{1, \ldots, n\} \to \{1, \ldots, n\}$), write $\sigma\Gamma$ for the sequent $A_{\sigma 1}, \ldots, A_{\sigma n}$. Proofs are generated using the rules in Figure 2. As a technical convenience, we shall often supress permutation (perm) rules, for example, writing

$$\frac{\Gamma,\ A,\ B, \Delta}{\Gamma,\ A \parr B, \Delta}\ \parr$$

which leaves implicit a permutation rule above and below the $\parr$-rule, if Δ is non-empty.

2.2 Coherence spaces

We write $\frown$ for strict coherence and $\#$ for strict incoherence of coherence spaces [Gir87, §3]. We call $\frown$ ***adjacency*** and $\#$ ***conflict***. The elements of the web $|X|$ of a coherence space X are ***tokens*** of X. Recall that a ***map*** $X \to Y$ between coherence spaces is a binary relation $R \subseteq |X| \times |Y|$ which preserves strict coherence and reflects strict incoherence: $y_1 R^{\mathrm{op}} x_1 \frown x_2 R y_2$ implies $y_1 \frown y_2$, and $x_1 R y_1 \# y_2 R^{\mathrm{op}} x_2$ implies $x_1 \# x_2$. (We write $x R y$ or $y R^{\mathrm{op}} x$ for $\langle x, y \rangle \in R$.)

3 Conflict linkings

Informal definition. A *link* on a sequent Γ is an edge between dual leaves. A *linking* on Γ is a finite set L of links on Γ equipped with a symmetric, irreflexive binary *conflict* relation $\# \subseteq L \times L$

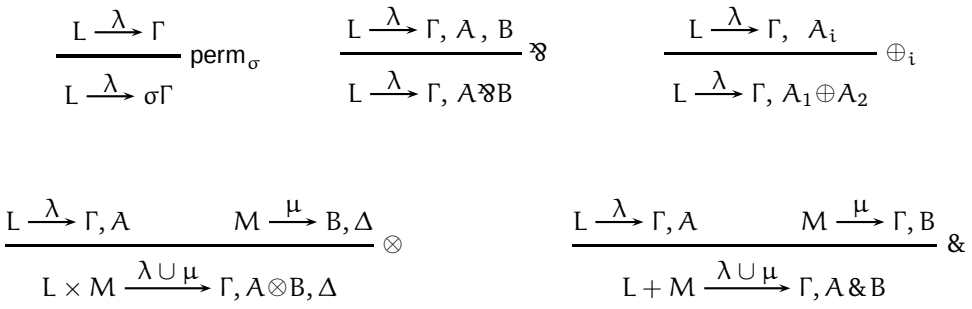


Figure 3: Inductive translation from a proof to a conflict linking.

such that overlap implies conflict: if distinct links l and m share an atom, then $l \# m$. Links may be parallel (between the same pair of leaves). Examples of linkings are shown in Figure 1. When drawing linkings, we leave implicit the conflicts implied by overlap.

Formalisation. A ***dual pair*** in Γ is a pair $\{x, y\}$ of dual leaves in Γ.

DEFINITION 1 *A **linking** on Γ is a binary relation $\lambda : L \to |\Gamma|$ from a finite coherence space L, whose tokens are called **links** on Γ, to the set $|\Gamma|$ of leaves in Γ, satisfying:*

- ***Dual pair.*** *For every link l in L the direct image $\lambda[l] = \{x \in |\Gamma| : \langle l, x\rangle \in \lambda\}$ is a dual pair.*
- ***Overlap.*** *If $\langle l, x\rangle \in \lambda$ and $\langle l', x\rangle \in \lambda$ with $l \neq l'$ (l and l' **overlap** at x) then $l \# l'$ [mod L].*

We abbreviate a linking $\lambda : L \to |\Gamma|$ to $\lambda : L \to \Gamma$ or $L \xrightarrow{\lambda} \Gamma$.

4 P-time proof translation function from proofs

Informal definition. A MALL proof of Γ translates to a linking on Γ by viewing each axiom rule as a link on Γ (by tracing its two leaves down the proof into Γ), and defining $l \# m$ iff l and m are in opposite branches above a &-rule. Figure 1 shows examples of proof translation.

Formalisation. The following formalisation is by induction on the number of rules in a proof.

- *Base case.* The axiom rule $\dfrac{}{P,\ \overline{P}}$ translates to the unique single-link linking on $P, \overline{P}$.[11]
- *Inductive step.* Every instance of a rule induces an inclusion function from the leaves of each sequent above the line to the sequent below the line.[12] Via these leaf inclusions, Figure 3 interprets each rule as an operation on linkings. The sum $L + M$ in the interpretation of the &-rule is the disjoint union (categorical sum/coproduct) of the coherence spaces L and M, denoted $L \oplus M$ in [Gir87]. Without loss of generality, we assume the canonical injections from the token sets of L and M into the token set of $L + M$ are inclusions. The product $L \times M$ in the interpretation of the $\otimes$-rule is $L + M$ together with strict coherence between every token in L and every token in M. This is categorical product, denoted $L \& M$ in [Gir87].

[11] If x and x' are the two leaves, the linking is $\lambda : I \to P, \overline{P}$ where I has a single token $\bullet$ and $\lambda = \{\langle \bullet, x\rangle, \langle \bullet, x'\rangle\}$.
[12] Each sequent (parse forest) above the line is a subgraph of the sequent below the line.

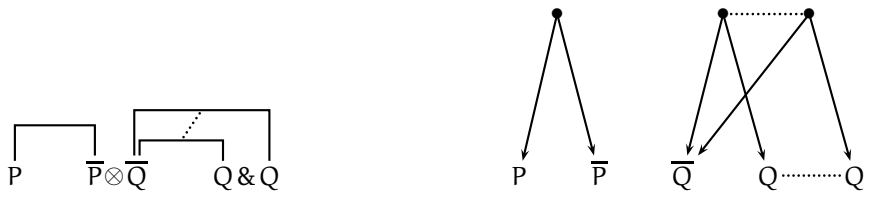


Figure 4: An example of a slicing $\lambda : L \to \Gamma$ with $\Gamma = P, \overline{P} \otimes \overline{Q}, Q, Q$. The underlying maximal map $\lambda : L \to \Gamma^{\#}$ between contractible coherence spaces is shown on the right. The fives tokens of the coherence space $\Gamma^{\#}$ are shown with their literal labels. The three links (tokens) of L are shown as $\bullet$. Note that λ is indeed maximal: were we to add any edge to the binary relation λ, it would no longer be a coherence space map.

Each rule interpretation preserves the *Dual pair* and *Overlap* conditions in the definition of a linking. Thus the translation of a proof is a well-defined linking.

A linking is ***sequentializable*** if it is the translation of a proof; any such a proof is a ***sequentialization*** of the linking.

5 Slicings

This section defines a *slicing* as a refinement of a linking, a stepping stone towards the definition of conflict net.

A coherence space is ***contractible*** if its web is finite and P_4-free (no induced four-vertex path [Sei74]): whenever $x_1 \# x_2 \# x_3 \# x_4$ for distinct x_i then $x_1 \# x_3$ or $x_2 \# x_4$ or $x_1 \# x_4$ [Hu99]. Define $\Gamma^{\#}$ as the coherence space whose tokens are the leaves of Γ with $x \# y$ iff $x \neq y$ and the smallest subformula containing x and y is additive.[13] If Γ is non-empty, its coherence space $\Gamma^{\#}$ is contractible (a simple induction).

DEFINITION 2 *A **slicing** on Γ is a maximal map $\lambda : L \to \Gamma^{\#}$ from a contractible coherence space L.*

Maximality is with respect to inclusion among maps $L \to \Gamma^{\#}$.[14] An example of a slicing is shown in Figure 4 with its underlying maximal map clarified.

PROPOSITION 1 *Checking that a linking $\lambda : L \to \Gamma$ is a slicing is p-time in the sizes of L and Γ.*

Proof. Checking that λ is a map (preserving $\frown$ and reflecting $\#$) is clearly polynomial. Checking contractibility (P_4-freeness) is linear [CPS85]. Checking direct images are dual pairs is obviously polynomial. Checking maximality is polynomial: for every edge $e \notin \lambda$ we check that $\lambda \cup \{e\}$ is not a map.[15] □

A ***slice*** of a slicing $\lambda : L \to \Gamma^{\#}$ is a maximal clique in L.[16] The two slices of the example in Figure 4

[13] In other words, $x \# y$ iff x and y are in the same formula A, and the first common vertex along the paths from x and y to the root of A is labelled & or $\oplus$. Equivalently, the join (least upper bound) z of x and y exists when we interpret Γ as a partial order with leaves maximal and roots minimal, and z is labelled & or $\oplus$.

[14] Thus λ is maximal iff it is a maximal clique in $L \multimap \Gamma^{\#}$.

[15] It suffices to test with single extra edges e since a map $R : X \to Y$ is maximal iff it is a maximal clique in the coherence space $X \multimap Y$.

[16] A clique C is a set of pairwise coherent tokens: if $x, y \in C$ and $x \neq y$ then $x \frown y$.

are illustrated below.

An ***additive resolution*** of Γ is a maximal clique in $\Gamma^{\#}$ [HG03, HG05]. The ***image*** of a set $Z \subseteq X$ under a binary relation $R \subseteq X \times Y$ is $\{y \in Y : zRy \text{ for some } z \in Z\}$. The following proposition formalises the sense in which "every slice is an MLL linking" (*cf.* [Gir87, Gir96, HG03, HG05]).

PROPOSITION 2 *Let $\lambda : L \to \Gamma$ be a non-empty slicing. The image of every slice of λ is an additive resolution of Γ.*

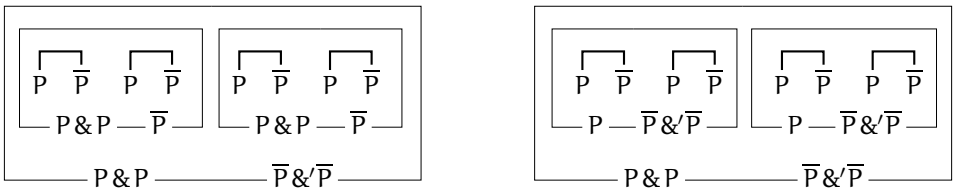
Proof. A corollary of [Hu99, Prop. 2.2]: a non-empty map between contractible coherence spaces is maximal iff it preserves maximal cliques, *i.e.*, the image of any maximal clique is a maximal clique. □

Note that the proposition holds for the two slices depicted above. The proposition is somewhat surprising, since checking every slice appears exponential-time (because a slice is a subset).

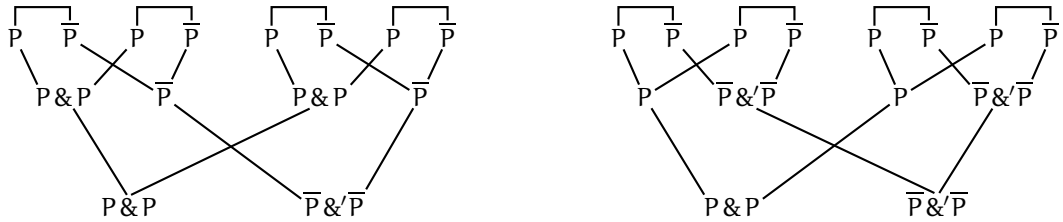
6 Introducing erasure: Boxless nets

In Section 7 we define a conflict net as a slicing which is *erasable* under a confluent, terminating (strongly normalising) *erasure* rewrite $\rightsquigarrow$. Erasability is checkable in p-time in the number of links and in the number of leaves in the sequent. A form of erasure will also yield p-time correctness for the slice nets of [HG03, HG05]. For didactic purposes, we begin by defining erasure in a simple setting related to box nets [Gir87], since that is the most likely to be familiar to the reader. However, the reader can safely skip to Section 7 without loss of continuity.

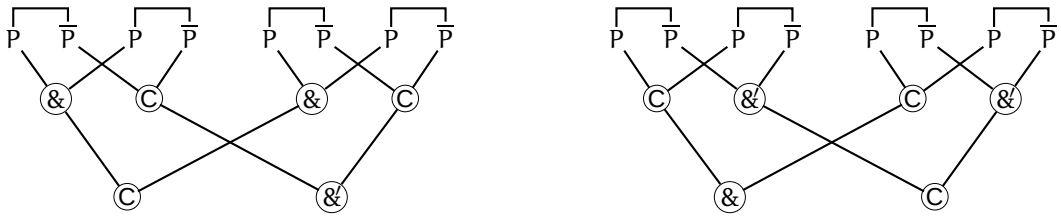
We shall describe a variant of box nets in which the circumscribing boxes are not drawn explicitly. Accordingly, we shall refer to them as *boxless nets*. The translation from a proof to a boxless net is exactly the same as the translation to a box net — only one forgets to draw the boxes. For example, the two proofs on page 1 translate (respectively) to the following pair of box nets:



Now emphasise the superposition/contraction of these formulas, and drop the surrounding boxes:



Finally, draw nodes instead of formulas, to remove some redundancy, and where two formulas merge, make that explicit by drawing a contraction node (C-node):



6.1 Circuits

A ***circuit*** comprises:

- A finite, non-empty set of ***nodes***.
- A finite set of ***wires***. Each wire is labelled with a formula, and is assigned a ***source*** node and, possibly, a ***target*** node. If a target node is present, it is distinct from the source node. A wire with no target is an ***exit***.
- Each node has one of the following forms:
  - ***Axiom***. The source of two wires and the target of none. The wires are labelled by dual literals.[17]
  - ***Contraction***. The target of two wires and the source of one. All three wires have the same formula.
  - ***Binary***. The target of two wires and the source of one. The incoming wires are distinguished as a ***left*** wire and a ***right*** wire. A binary node is typed as one of $\otimes$, $\parr$ or $\&$. If the formula of the left wire is A, the formula of the right wire is B, and the node type is $\Box$, the formula of the outgoing wire is $A \Box B$.
  - ***Plus***. The target of one wire and the source of one wire. The incoming wire is distinguished as ***left*** or ***right***. Let A be the formula of the incoming wire. If the incoming wire is left (resp. right) then the formula of the outgoing wire is $A \oplus B$ (resp. $B \oplus A$) for some formula B.
- The graph is connected: for any two nodes N and N' there exists a sequence of nodes $N_1 \dots N_k$ with $N_1 = N$ and $N_k = N'$ ($k \geqslant 1$) such that for all $i \in \{1, \dots, k-1\}$ the nodes N_i and N_{i+1} are joined by a wire, *i.e.*, there exists a wire whose source is N_i and target is N_{i+1}, or vice versa.[18]
- The exits are equipped with a linear order. The sequent comprising the formulas of the exits, in order, is the ***conclusion*** of the circuit.

An example of a circuit with concluding sequent $P \& P, \overline{P} \& \overline{P}$ is drawn in Figure 5, formalising the last graph in our motivating discusion above. An axiom node is drawn as a horizontal line segment.

[17] If we wish to include cuts, we define a cut node as the target of two wires, labelled by dual formulas, and the source of no wire.

[18] By dropping connectedness, and slightly modifying the definition of erasure below, one could choose to validate the mix rule.

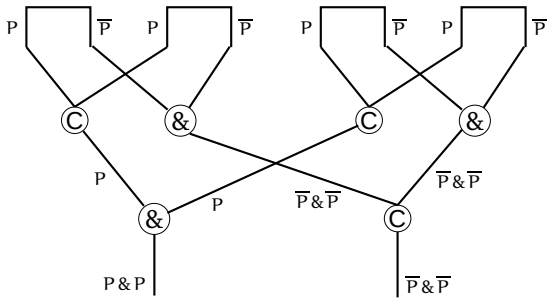


Figure 5: An example of an erasable circuit (boxless net).

Wires are oriented downwards in the page (*i.e.*, the target of a wire, when present, is below its source). Left/right incoming wires are distinguished by their contact point being left/right of the centre of the target node. Contraction nodes are marked C. Each wire is labelled with its formula. The exits are ordered from left to right in the page. (The style is similar to interaction nets [Laf90].)

6.2 Erasure

A node is ***final*** if it is the source of an exit wire. A node N is ***ready*** if it is final and it matches one of the following cases:

- N is a $\oplus$.
- N is a $\otimes$. Deleting N, and its exit wire, disconnects the circuit (*i.e.*, the result of deleting N is a disjoint union of two connected components).
- N is a $\parr$. Deleting N, and its exit wire, does not disconnect the circuit.
- N is a $\&$. Every other final node is a contraction-node. Deleting all final nodes, and their exit wires, yields exactly two connected components X_1 and X_2. Every final node in the original circuit has one incoming wire in X_1 and the other in X_2.
- N is an axiom-node, the unique node of the circuit.

Write $X \rightsquigarrow_N S$ if S is the set of connected components resulting from deleting the ready node N, each promoted to a circuit by adding the exit-order induced canonically from the exit-order of X. By definition of readiness:

- if N is a $\oplus$ or $\parr$ then $S = \{X'\}$, a single circuit,
- if N is a $\otimes$, $\&$ or cut-node, then $S = \{X_1, X_2\}$, two circuits.
- if N is an axiom-node, then $S = \emptyset$, the empty set.

If T and U are sets of circuits, write $T \rightsquigarrow_{X,N} U$ if $T = T' \cup \{X\}$ (disjoint union), $X \rightsquigarrow_N S$, and $U = T' \cup S$. (In other words, we replace X by the circuit(s) resulting from deleting N from X.) Write $T \rightsquigarrow U$ if $T \rightsquigarrow_{X,N} U$ for some X and N. Note that X and N are uniquely determined given T and U; we call N the ***redex***. The relation/rewrite $\rightsquigarrow$ on sets of circuits is called ***erasure***.

PROPOSITION 3 *Erasure $\rightsquigarrow$ satisfies the diamond property: if $T \rightsquigarrow U_0$ and $T \rightsquigarrow U_1$ with $U_0 \neq U_1$, there exists V such that $U_0 \rightsquigarrow V$ and $U_1 \rightsquigarrow V$.*

Proof. Suppose $T \rightsquigarrow_{X_i, N_i} U_i$. Assume $X_0 = X_1$, or else the result is immediate. Let $X = X_0 = X_1$. Necessarily $N_0 \neq N_1$ (otherwise $U_0 = U_1$), therefore N_i cannot be a &-node (since if a &-node is a redex, there can be no other redex in the same circuit), and cannot be an axiom-node. Without loss of generality, ignore cut-node redexes, since they are homologous to $\otimes$-redexes. Thus we are left to consider the following node-types for the redexes N_0 and N_1: $\parr$, $\oplus$, $\otimes$. The diamond property is then immediate, since each reduction in these cases merely deletes a single vertex from a graph. □

Due to more abstract superposition, erasure on conflict nets will not satisfy the diamond property. (It will nonetheless be confluent.)

PROPOSITION 4 *Erasure $\rightsquigarrow$ is terminating (strongly normalising).*

Proof. If $U \rightsquigarrow V$ then the disjoint union of the circuits in V has strictly less nodes than the disjoint union of the circuits in U. □

Write $\rightsquigarrow^*$ for the transitive closure of erasure $\rightsquigarrow$.

PROPOSITION 5 *Erasure $\rightsquigarrow$ is confluent: if $T \rightsquigarrow^* U_0$ and $T \rightsquigarrow^* U_1$ then there exists V such that $U_0 \rightsquigarrow^* V$ and $U_1 \rightsquigarrow^* V$.*

Proof. Cut elimination is locally confluent (since it has the diamond property) and is terminating, so confluence follows from Newman's lemma [New42]. □

Thus every set of circuits has a unique $\rightsquigarrow$-normal form. A set of circuits S is ***erasable*** if its normal form is empty, *i.e.*, if $S \rightsquigarrow^* \emptyset$. A circuit X is erasable if $\{X\}$ is erasable.

DEFINITION 3 *A **boxless net** is an erasable circuit.*

Figure 5 depicts an example of a boxless net X. An erasure sequence for X is illustrated in Figure 6. Note that, by the diamond property, any erasure sequence from X to $\emptyset$ has the same number of steps: the number of non-contraction nodes in X.

6.3 P-time correctness

The following theorem distinguishes erasability from mere sequentializability.

THEOREM 1 *Erasability of a circuit X can be checked in p-time in the number of nodes in X.*

Proof. Let k be the number of nodes in X, and n the number of non-contraction nodes. Since each erasure step deletes a non-contraction node, the $\rightsquigarrow$-normal form of $\{X\}$ is obtained in at most n steps. By the diamond property, any ready node N suffices at each step. To find such an N requires checking at most n nodes for readiness, and the complexity of checking if a node is ready is at worst the complexity of checking disconnectedness of a graph G into two connected components, where G has at most k vertices. □

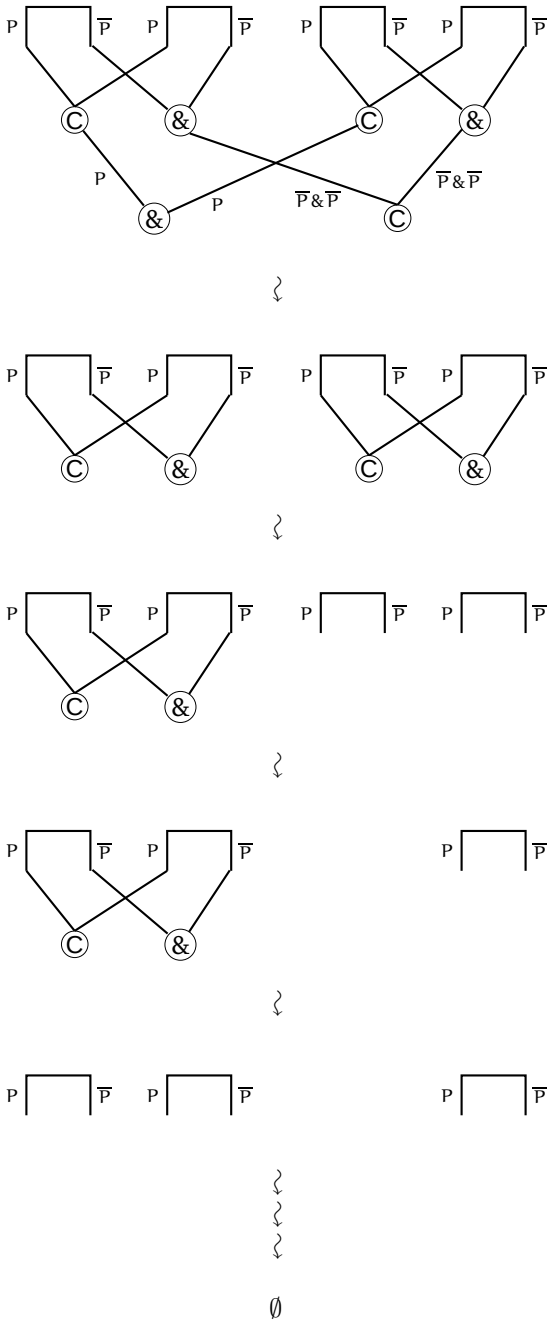


Figure 6: An erasure sequence. To save space, we leave exit wires from final & and C nodes implied.

6.4 Translation function from proofs to circuits

The obvious translation via box nets was outlined at the beginning of the section: simply forget to draw the boxes. For the sake of complete rigour, we give below a direct formal translation of a proof Π to a circuit X, by induction on the number of rules in Π.

- *Base case.* Π is an axiom with conlusion $P, \overline{P}$. X is an axiom-node N two exit wires, labelled P and $\overline{P}$, in that order.
- *Induction step.* Let ρ be the last rule of Π, and Γ its conclusion.
  - *Unary case.* ρ has one hypothesis sequent Δ above its line, which concludes the subproof Π' of Π. Let X' be the circuit obtained from Π'.
    - $\rho = \mathsf{perm}_\sigma$. Define X from X' by applying the permutation σ to the ordering of the exit wires (viewing the ordering as an enumeration from 1).
    - $\rho = \parr$, so $\Delta = \Delta', A, B$ and $\Gamma = \Delta', A \parr B$. Define X from X' as follows: add a new $\parr$-node N as the target of the last two exit wires of X' (the last wire being designated *right* for N); add to N a new exit wire w labelled $A \parr B$; place w in last position in the exit wire order.
    - $\rho = \oplus_i$, so $\Delta = \Delta', A_i$ and $\Gamma = \Delta', A_0 \oplus A_1$. Define X from X' as follows: add a new $\oplus$-node N as the target of the last wire v of X', and designate v as left or right according to $i = 0$ or 1; add to N a new exit wire w labelled $A_0 \parr A_1$; place place w in last position in the exit wire order.
  - *Binary case.* ρ has two hypotheses Δ_0 and Δ_1, which conclude subproofs Π_0 and Π_1 of Π, respectively. Let X_i be the circuit obtained from Π_i.
    - $\rho = \otimes$, so $\Delta_0 = \Delta_0', A$ and $\Delta_1 = B, \Delta_1'$. Define X from the disjoint union of X_0 and X_1 as follows: add a new $\otimes$-node N as the target of the last wire v_0 of X_0 and the first wire v_1 of X_1; designate v_0 as left for N and v_1 as right; add to N a new exit wire w labelled $A \otimes B$; impose the following order on exit wires: all the exit wires of X_0 in their original order (except v_0, which is no longer an exit), then w, then all the exit wires of X_1 in their original order (except v_1, which is no longer an exit).
    - $\rho = \&$, so $\Delta_i = \Delta', A_i$. Let $\Delta' = B_1, \ldots, B_n$. Define X from the disjoint union of X_0 and X_1 as follows: add a new $\&$-node N as the target of the last wire v_0 of X_0 and the last wire v_1 of X_1; designate v_0 as left for N and v_1 as right; add to N a new exit wire w labelled $A \otimes B$; for $j = 1, \ldots, n$ add a new contraction-node N_j as the target of the j^{th} wire of X_0 and the j^{th} wire of X_1; add to N_j a new exit wire w_j labelled B_j; impose the following order on exit wires: $w_1, \ldots, w_n, w$.

PROPOSITION 6 *The above translation maps every proof to an erasable circuit.*

Proof. By induction on the number of rules in the proof Π. We reference each case in the translation above:

- *Base case.* X is erasable in one step: $\{X\} \rightsquigarrow_{X,N} \emptyset$.
- *Induction step.*

– $\rho = \mathsf{perm}_\sigma$. The circuits X and X' differ only in the order on their exit wires. Since node readiness is independent of exit wire order, X is erasable by the same sequence of erasures as X'.

– $\rho = \parr$ or $\oplus$. $\{X\} \rightsquigarrow_{X,N} \{X'\}$ by construction, and X' is erasable.

– $\rho = \otimes$ or $\oplus$. $\{X\} \rightsquigarrow_{X,N} \{X_1, X_2\}$ by construction, and each X_i is erasable. Thu X is erasable by (arbitrarily) interleaving erasure sequences of X_1 and X_2 after $\{X\} \rightsquigarrow_{X,N} \{X_1, X_2\}$. □

A circuit X is ***sequentializable*** if it is the translation of a proof; any such proof is a ***sequentialization*** of X.

6.5 Sequentialization

THEOREM 2 (SEQUENTIALIZATION) *A circuit is erasable iff it is sequentializable.*

Proof. The right-to-left implication is Proposition 6.

Let X be an erasable circuit, with n-step erasure sequence to $\emptyset$. We prove X sequentializable by induction on n (which is the same for all erasure sequences to $\emptyset$, by the diamond property).

- *Base case.* $n = 1$. X is the translation of an axiom rule.
- *Inductive step.* $n > 1$. Let N be the ready node deleted from X in the first erasure step. Let $v_1, \ldots, v_n$ be the exit wires of X, in order, and let C_i be the formula of v_i. Suppose v_k be the exit wire of N ($1 \leqslant k \leqslant n$) and let $\Gamma_1 = C_1, \ldots, C_{k-1}$ and $\Gamma_2 = C_{k+1} \ldots C_n$. We split into subcases according to the type of N.
  - *Unary case.* N is a $\parr$ or $\oplus$. Thus $\{X\} \rightsquigarrow_{X,N} \{Y\}$ is the first erasure step. By induction hypothesis, a proof Π translates to Y.
    * N is a $\parr$. Let A be the formula of the left incoming wire of N, and B the formula of the right incoming wire. The following proof translates to X:

$$\dfrac{\dfrac{\Pi}{\Gamma_1, A, B, \Gamma_2}}{\Gamma_1, A \parr B, \Gamma_2}\,\parr$$

      (Permutation rules are supressed; see Section 2.1.)
    * N is a $\oplus$. Thus the formula C_k of N's exit wire v_k is $A_0 \oplus A_1$. The following proof translates to X, where $j = 0/1$ according as the incoming wire of N is designated left/right.

$$\dfrac{\dfrac{\Pi}{\Gamma_1\ ,\ A_i\ ,\ \Gamma_2}}{\Gamma_1, A_0 \oplus A_1, \Gamma_2}\,\oplus_i$$

  - *Binary case.* N is a $\otimes$ or $\&$. Thus $\{X\} \rightsquigarrow_{X,N} \{Y_0, Y_1\}$ is the first erasure step. By induction hypothesis, proofs Π_i translate to Y_i. Let u_0 be the left incoming wire of N, labelled A_0, and u_1 its right incoming wire, labelled A_1.

* N is a $\otimes$. Thus $C_k = A_0 \otimes A_1$. The conclusion of Π_i is Δ_i, A_i, Δ'_i. The following proof translates to X:

$$\dfrac{\dfrac{\dfrac{\Pi_0}{\Delta_0, A_0, \Delta'_0}}{\Delta_0, \Delta'_0, A_0}\,\mathsf{perm} \qquad \dfrac{\dfrac{\Pi_1}{\Delta_1, A_1, \Delta'_1}}{A_1, \Delta_1, \Delta'_1}\,\mathsf{perm}}{\dfrac{\Delta_0, \Delta'_0, A_0 \otimes A_1, \Delta_1, \Delta'_1}{\Gamma_1, A_0 \otimes A_1, \Gamma_2}\,\mathsf{perm}}\,\otimes$$

The permutations are determined by the fact that the exit wires of Y_0 and Y_1 apart from u_0 and u_1 are exactly the exit wires of X apart from w_k.

* N is a &. Thus $C_k = A_0 \,\&\, A_1$. The conclusion of Π_i is Γ_1, A_i, Γ_2. The following proof translates to X:

$$\dfrac{\dfrac{\dfrac{\Pi_0}{\Gamma_1, A_0, \Gamma_2}}{\Gamma_1, \Gamma_2, A_1}\,\mathsf{perm} \qquad \dfrac{\dfrac{\Pi_1}{\Gamma_1, A_1, \Gamma_2}}{\Gamma_1, \Gamma_2, A_1}\,\mathsf{perm}}{\dfrac{\Gamma_1, \Gamma_2, A_0 \,\&\, A_1}{\Gamma_1, A_0 \,\&\, A_1, \Gamma_2}\,\mathsf{perm}}\,\&$$

The permutations are determined by the bijections between the exit wires of each Y_i and the exit wires of X. □

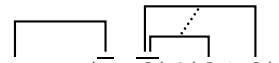
6.6 Relationship with contractibility/retractability

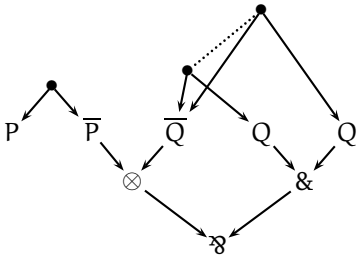
The underlying data structure of a circuit (aside from the order on exit wires, which is a technical convenience) is the same as that used by Maieli [Mai07].

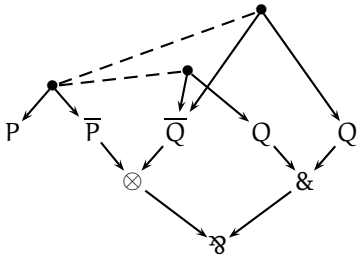
CONJECTURE 1 *A circuit is the translation of a proof iff it is retractable with respect to Maieli's* $R_1, \ldots, R_4$ *(dropping* R_5*).*

7 Erasure for conflict nets

We can draw a linking $\lambda : L \to \Gamma$ as a graph in two different ways, depending on whether we show conflict # or adjacency $\frown$. For example, the linking below is followed by each of its graphs, the former graph showing conflict # (dotted), the latter showing adjacency $\frown$ (dashed). The three links are shown as • vertices.







We shall write $\lambda^{\#}$ for the left graph, and $\lambda^{\frown}$ for the right graph. Formally,

$$\begin{aligned} \lambda^{\#} &= \Gamma \cup L^{\#} \cup \lambda \\ \lambda^{\frown} &= \Gamma \cup L^{\frown} \cup \lambda \end{aligned}$$

where $L^{\#}$ (resp. $L^{\frown}$) denotes the undirected graph on the links of L given by conflict (resp. adjacency), and (without loss of generality) we assume Γ and L are disjoint. Thus $\lambda^{\#}$ is the union of the sequent Γ (formula parse trees) and the $\#$-graph of L, together with an edge $l \to x$ whenever $\langle l, x\rangle \in \lambda$ (*i.e.*, whenever x is a leaf in the dual pair of l).

A vertex in a sequent with no outgoing edge is a ***root***, and is said to be ***final***. Let $\diamond \in \{\&, \oplus\}$ and let r be the $\diamond$-labelled root of the formula $A_0 \diamond A_1$ in Γ. A slicing $\lambda : L \to \Gamma$ ***touches*** A_i if some leaf of A_i is in the image of λ, and ***chooses*** A_i if it touches A_i but does not touch A_{1-i}. (Since λ is a slicing, if it is non-empty it must touch at least one of A_0 and A_1 by Proposition 2; it is possible that λ touches both.) If λ touches exactly one of the A_i we say that r is ***unary*** under λ. A ***piece*** of λ is its restriction to a connected component[19] of the $\frown$-graph $L^{\frown}$ of L. A slicing $\lambda : L \to \Gamma$ is ***connected*** if it is non-empty and its $\#$-graph $\lambda^{\#}$ is connected.

Let $\lambda : L \to \Gamma$ be a connected slicing. A $\Box$-labelled root r is ***ready*** in λ if one of the following cases holds:

- $\Box = \otimes$ and r is not in a cycle in $\lambda^{\#}$.[20]
- $\Box = \parr$.
- $\Box = \oplus$ and r is unary under λ.
- $\Box = \&$ and r is unary under every piece of λ.

Let $A_0 \Box A_1$ be the formula whose root is r. The result of ***erasing*** r, if r is ready, is a set of slicings $\lambda \setminus r$:

- $\Box = \otimes$. Let $\lambda_0^{\#}$ and $\lambda_1^{\#}$ be the connected components of $\lambda^{\#}$ upon deleting r. This yields two slicings λ_0 and λ_1, the former on a sequent Δ_0, A_0, Δ_0' and the latter on Δ_1, A_1, Δ_1'. Define $\lambda \setminus r = \{\lambda_0, \lambda_1\}$.

[19] By convention, a connected component is non-empty.

[20] In other words, upon deleting r (and its two incoming edges) there are two connected components.

- $\square = \parr$. Let ${\lambda_0}^{\#}$ be the result of deleting r from $\lambda^{\#}$, yielding a slicing λ_0. Define $\lambda \setminus r = \{\lambda_0\}$.
- $\square = \oplus$. Since r is unary under λ and λ is non-empty, λ chooses A_j for some $i \in \{0,1\}$. Let $\lambda_j^{\#}$ be the result of deleting r and A_{1-j} from $\lambda^{\#}$, yielding a slicing λ_j. Define $\lambda \setminus r = \{\lambda_j\}$.
- $\square = \mathbin{\&}$. Let $\Gamma = \Delta, A_0 \mathbin{\&} A_1, \Sigma$. Let λ_i be the slicing on Δ, A_i, Σ comprising the union of all pieces of λ which choose A_i. Define $\lambda \setminus r = \{\lambda_0, \lambda_1\}$. (By Proposition 2, every piece of λ chooses one of the A_i. Thus $\lambda = \lambda_0 \cup \lambda_1$.)

Note that even though λ is connected, a slicing in $\lambda \setminus r$ may be disconnected (*e.g.* empty).

A ***cluster*** is either a set of slicings or the ***error*** symbol E. Define ***erasure*** $\rightsquigarrow$ on clusters as follows.

- $Y \rightsquigarrow \mathsf{E}$ if Y contains a slicing which is disconnected. (Note: any empty slicing is disconnected.)
- $X \cup \{\lambda\} \rightsquigarrow X \cup (\lambda \setminus r)$ if r is a ready root of λ, and every slicing in X is connected. Here we assume $\lambda \notin X$.
- $X \cup \{\lambda\} \rightsquigarrow X$ if λ is a single link on $P, \overline{P}$ for some literal P (*i.e.*, if λ corresponds to an axiom), and every slicing of X is connected. Here we assume $\lambda \notin X$.

Write $\rightsquigarrow^*$ for the transitive closure of $\rightsquigarrow$.

PROPOSITION 7 *Erasure $\rightsquigarrow$ is locally confluent (weak Church-Rosser): if $X \rightsquigarrow Y_0$ and $X \rightsquigarrow Y_1$ there exists a cluster Z such that $Y_0 \rightsquigarrow^* Z$ and $Y_1 \rightsquigarrow^* Z$.*

Proof. Suppose $X \rightsquigarrow Y_i$ by erasing r_i from $\lambda_i \in X$. Assume $\lambda_0 = \lambda_1$, or else the result is immediate. Let $\lambda = \lambda_0 = \lambda_1$. Assume $r_0 \neq r_1$, otherwise the result holds with $Z = Y_0 = Y_1$. Let $\square_i$ be the connective of r_i. We split cases according to $\square_0$.

- $\square_0 = \otimes$. Let $\lambda \setminus r_0 = \{\lambda_a, \lambda_b\}$, with both λ_a and λ_b connected. Without loss of generality, assume r_1 is in the sequent of λ_a. We split cases according to $\square_1$.
  - $\square_1 = \oplus$ or $\parr$. Then $\lambda_a \setminus r_1 = \{\lambda'_a\}$. If λ'_a is disconnected (case $\square = \parr$ only), take $Z = \mathsf{E}$; otherwise define Z by replacing λ in X with $\{\lambda'_a, \lambda_b\}$.
  - $\square_1 = \otimes$. Then $\lambda_a \setminus r_1 = \{\lambda'_a, \lambda''_a\}$, with λ'_a and λ'_b connected. Define Z by replacing λ in X with $\{\lambda'_a, \lambda''_a, \lambda_b\}$.
  - $\square_1 = \mathbin{\&}$. Since r_0 is ready in λ, and λ is non-empty, λ must have a single piece. Thus r_1 is unary, so one of the two slicings obtained by removing r_1 is empty. Since r_1 remains unary after erasing r_0, we can take $Z = \mathsf{E}$.
- $\square_0 = \mathbin{\&}$. By r_0/r_1 symmetry, we need not consider $\square_1 = \otimes$. Let $\lambda \setminus r_0 = \{\lambda_a, \lambda_b\}$. Assume λ_a and λ_b are connected, or else the result is trivial with $Z = \mathsf{E}$. We consider subcases for $\square_1$.
  - $\square_1 = \mathbin{\&}$. Since there is no constraint on &-readiness, we can erase the &'s in either order. However, due to duplication, there are two copies of the second & to erase. Let Γ_a and Γ_b be the sequents of λ_a and λ_b. The sequents have copies r_{1a} and r_{1b} of r_1, respectively. We have $\lambda_a \setminus r_{1a} = \{\lambda_{ax}, \lambda_{ay}\}$ and $\lambda_b \setminus r_{1b} = \{\lambda_{bx}, \lambda_{by}\}$. Let $\lambda \setminus r_1 = \{\lambda_x, \lambda_y\}$. Analogously, $\lambda_x \setminus r_{0x} = \{\lambda_{xa}, \lambda_{xb}\}$ and $\lambda_y \setminus r_{0y} = \{\lambda_{ya}, \lambda_{yb}\}$. Since &-removal merely partitions the pieces of λ, we have $\lambda_{ax} = \lambda_{xa}$, and similarly for the other three. If any

of the four slicings is empty, we take $Z = \mathsf{E}$. Otherwise, let $X = X' \cup \{\lambda\}$, where $\lambda \notin X'$. Define $Z = X' \cup \{\lambda_{ax}, \lambda_{ay}, \lambda_{bx}, \lambda_{by}\}$. Then

$$X \rightsquigarrow_{r_0} Y_0 \rightsquigarrow_{r_{1a}} X' \cup \{\lambda_{ax}, \lambda_{ay}, \lambda_b\} \rightsquigarrow_{r_{1b}} Z$$
$$X \rightsquigarrow_{r_1} Y_1 \rightsquigarrow_{r_{0x}} X' \cup \{\lambda_{xa}, \lambda_{xb}, \lambda_y\} \rightsquigarrow_{r_{0y}} Z$$

where the $\rightsquigarrow$-subscripts indicate which root is being erased.

– $\Box_1 = \oplus$ and $\parr$. The reasoning is analogous to the previous case, though simpler due to less duplication.

- $\Box_0 = \parr$. By symmetry, we need only consider $\Box_1 = \parr$ or $\oplus$. This case is trivial, since erasing each r_i merely deletes a vertex from a (sequent)-graph. It is possible that erasing a $\parr$ can yield a disconnected slicing; in this case we take $Z = \mathsf{E}$.
- $\Box_0 = \oplus$. By symmetry, we need only consider $\Box_1 = \oplus$. This case is trivial.

If either Y_i is E we simply take $Z = \mathsf{E}$. □

Define the ***profile*** of a cluster as $\langle p, q\rangle$ where p is the total number of links (summed accross all slicings) plus the total number of conflict edges, and q is the total number of connectives (in the underlying sequents).

THEOREM 3 *Erasure $\rightsquigarrow$ is terminating (strongly normalising).*

Proof. Every $\rightsquigarrow$-step either (a) decreases p, while perhaps increasing q, or (b) decreases q, without increasing p. □

PROPOSITION 8 *Erasure $\rightsquigarrow$ is confluent: if $X \rightsquigarrow^* Y_0$ and $X \rightsquigarrow^* Y_1$ then there exists Z such that $Y_0 \rightsquigarrow^* Z$ and $Y_1 \rightsquigarrow^* Z$.*

Proof. Cut elimination is locally confluent and terminating, hence confluent by Newman's lemma [New42]. □

Thus every cluster has a unique $\rightsquigarrow$-normal form. A cluster X is ***erasable*** if its normal form is empty, *i.e.*, if $X \rightsquigarrow^* \emptyset$. A slicing λ is erasable if $\{\lambda\}$ is erasable.

DEFINITION 4 *A **conflict net** is an erasable slicing.*

7.1 P-time correctness

The ***size*** of a coherence space is its number of tokens, and the size of a sequent is its number of vertices.

THEOREM 4 *Erasability of a slicing $\lambda : L \to \Gamma$ can be checked in p-time in the sizes of L and Γ.*

Proof. Let $\{\lambda\} = X_0 \rightsquigarrow X_1 \rightsquigarrow \ldots \rightsquigarrow X_n$ be a normalisation sequence, let l be the size of L, and let g be the size of Γ. Let $m = l^2$, an upper bound on the number of conflict edges in L. Let $k = l + m$. Then $n \leqslant k.g$ since whenever a $\rightsquigarrow$-step decreases p in the profile $\langle p, q\rangle$, it increases q to at most g, and p remains at most k.

It remains to show that determining if a cluster X has a $\rightsquigarrow$-redex — and if so, executing the $\rightsquigarrow$-step — is p-time in l and g. First we check to see if every slicing in X is connected, which is p-time in the total number $\nu(X)$ of vertices in X, and $\nu(X) \leqslant gl + l$. (In the worst case, X has l slicings, each a single link on Γ.) If every slicing $\mu \in X$ is connected, we attempt to find a $\rightsquigarrow$-redex. Erasing axioms is trivial, therefore at worst we take each final vertex of X in turn, and check for readiness. Checking for readiness involves only finding connected components of graphs ($M^{\frown}$ and $\mu^{\#}$, where M is the domain of μ). □

7.2 Sequentialization

THEOREM 5 (SEQUENTIALIZATION) *A linking is a conflict net iff it is sequentializable.*

Proof. The right-to-left implication is a routine induction over the interpretation of rules as operations on linkings (Figure 3).

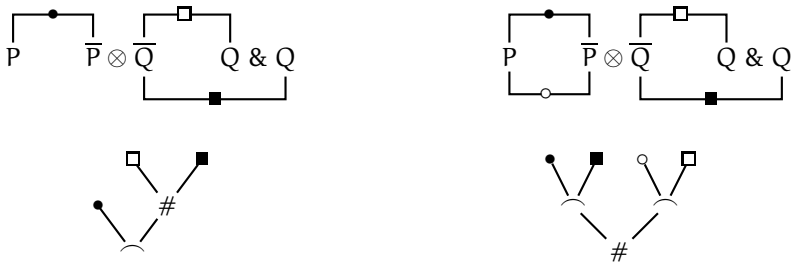
Conversely, a normalisation sequence $\{\lambda\} = X_1 \rightsquigarrow \ldots \rightsquigarrow X_n = \emptyset$ produces a proof rule-by-rule, from bottom-to-top, exactly as in the case of circuit nets (see the proof of Theorem 2). Every $\rightsquigarrow$-step yields one non-permutation rule, plus some permutations. □

8 Alternative representations of conflict nets

Translation from a proof to a conflict net is quadratic-time in the size of the proof (due to the conflict edges). If we are willing to code slightly more information in the representation, we can obtain a variant for which translation is linear time. A *sum net* collapses all parallel axiom links to a single link, and labels every axiom link with a formal sum of monomials. For example, here are the sum net representations of the two conflict nets at the bottom of Figure 1, respectively:

Girard discusses a relationship between monomials and coherence in Appendix A.1.1 of [Gir96].

A *tree net* is another alternative. The undirected graph of the # conflict relation of a proof net is always P_4-free (contractible), thus can be represented by a tree (the so-called *cotree* associated with a P_4-free graph). For example, here are the tree net versions of the last two conflict nets in Figure 1:



This tree on axiom links is obtained readily from a proof, in linear time: it is the underlying $\otimes$- and &-rule binary tree, modulo associativity and commutativity, with $\otimes$-rules providing strict coherence $\frown$ between axioms, and & providing conflict (strict incoherence) $\#$.

9 P-time correctness for slice nets, by erasure

By using erasure, we prove that the correctness of a slice net Λ on Γ [HG03, HG05] can be checked in p-time in the number of links in Λ and the number vertices in Γ. Recall that a linking of a slice net is a slicing $\lambda : L \to \Gamma$ with L a non-empty clique.

Let Λ be a set of linkings, or *linking-set*, on Γ. A link in/of Λ is a link in a linking of Λ (*i.e.*, a link in $\bigcup\Lambda$). Define $G(\Lambda, \Gamma)$ as the graph comprising Γ and every link in Λ. Λ is ***connected*** if it is non-empty and $G(\Lambda, \Gamma)$ is connected.

Let Λ be a connected linking on Γ, and let r be a root of Γ, the root of the formula $A_0 \square A_1$. Define r as ***ready*** if it matches one of the following cases:

- $\square = \parr$.
- $\square = \&$.
- $\square = \oplus$ and r is unary: for some $j \in \{0, 1\}$ no link in Λ has a leaf in the formula A_j.
- $\square = \otimes$. Deleting r disconnects G into two components G_i, where A_i is a formula in G_i. Let the underlying sequent of G_i be Δ_i. For each linking $\lambda \in \Lambda$ define λ_i as the restriction of λ to Δ_i (thus $\lambda = \lambda_0 \cup \lambda_1$). Define $\Lambda_i = \{\lambda_i : \lambda \in \Lambda\}$. Let n_i be the number of linkings in Λ_i, and n the number of linkings in Λ. Then[21]

$$n \quad = \quad n_0 \times n_1.$$

When ready, the result $\Lambda \setminus r$ of ***erasing*** r is:

- $\square = \parr$. Λ_0 on Γ_0, where Γ_0 has A_0, A_1 in place of $A_0 \parr A_1$.
- $\square = \oplus$. Λ_j on Γ_j, where Γ_j has A_j in place of $A_0 \oplus A_1$, according to whether a link of Λ has a leaf in A_j.

[21] By construction, $n \leqslant n_0 \times n_1$ always holds, since we work with *sets* of linkings.

- $\square = \&$. Λ_0 on Γ_0 and Λ_1 on Γ_1, where Γ_i has A_i in place of $A_0 \& A_1$, and Λ_i comprises every linking of Λ which has a link with a leaf in A_i. (Thus $\Lambda = \Lambda_0 \cup \Lambda_1$, disjointly.)
- $\square = \otimes$. Λ_0 on Δ_0 and Λ_1 on Δ_1, where Δ_i and Λ_i are as in the definition of $\otimes$-readiness above.

Note that even though Λ is connected, a linking-set in $\lambda \setminus r$ may be disconnected (*e.g.* empty).

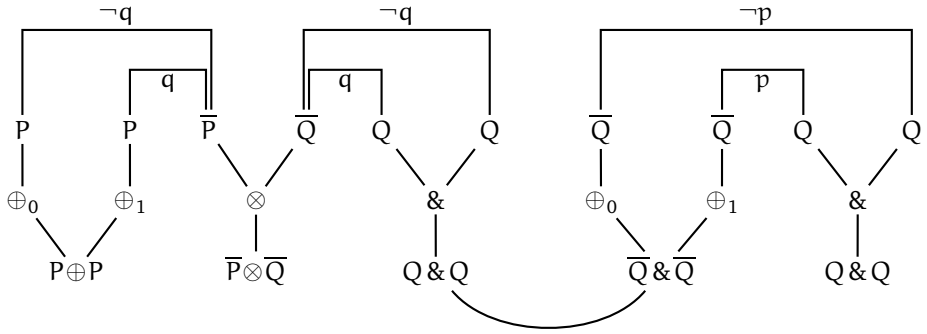
The following definitions are practically identical to those for erasure of conflict nets. A ***cluster*** is either a set of linking-sets or the ***error*** symbol E. Define ***erasure*** $\rightsquigarrow$ on clusters as follows.

- $Y \rightsquigarrow \mathsf{E}$ if Y contains a linking-set which is disconnected. (Note: any empty linking-set is disconnected.)
- $X \cup \{\Lambda\} \rightsquigarrow X \cup (\Lambda \setminus r)$ if r is a ready root of Λ, and every linking-set in X is connected. Here we assume $\Lambda \notin X$.
- $X \cup \{\Lambda\} \rightsquigarrow X$ if Λ has a single link, on $P, \overline{P}$ for some literal P (*i.e.*, if Λ corresponds to an axiom), and every linking-set of X is connected. Here we assume $\Lambda \notin X$.

Erasure $\rightsquigarrow$ is confluent and termining by the same reasoning as for conflict nets. The same reasoning with profiles shows that the path-length to normal form is polynomial in the number of links l and the number of sequent vertices g. Each form of readiness for a root is clearly p-time checkable. That erasure coincides with sequentializability is again a routine induction, as with circuits and conflict linkings.

10 Cut elimination

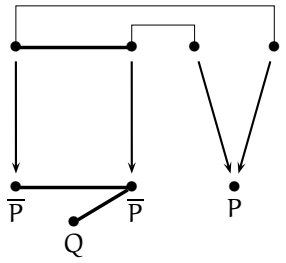
Cut elimination for conflict nets is work in progress. The same is true for monomial nets: the proposal for their cut elimination sketched in [Gir96, App. A.1.2–3] is ill-defined. A counter-example is shown below.



The definition of cut elimination fails to work because spreading is limited to a single formula: this means that after spreading above the central $Q \& Q$ with respect to p, we do not have a proof structure (contrary to the claim at the end of A.1.2 in [Gir96]). To fix cut elimination, one would at a minimum have to extend spreading: in the example above, performing something related to spreading above the left-most formula $P \oplus P$.

11 Relationship with combinatorial proofs

A *combinatorial proof* [Hug06a] is an abstraction notion of proof net for classical logic [Hug06b]. A combinatorial proof of a classical formula A is a graph homomorphism $h : L \to G(A)$ from a partitioned P_4-free (contractible) graph L to a graph $G(A)$ associated with A, satisfying certain conditions. A combinatorial proof of Peirce's law $((\overline{P} \vee Q) \wedge \overline{P}) \vee P$ is shown below.



The partitioned graph L is on top, with four vertices and one (thick, horizontal) edge, and two two-vertex classes indicated by (thin) link-style edges. The graph $G(A)$ is underneath, with four vertices and two edges. Its vertices are the literals of A, with an edge between literals when the smallest subformula containing them is a conjunction. The arrows indicate the graph homomorphism h.

The graph homomorphism is required to be a *skew fibration*. A coherence space map, as in a slicing, is just a relational generalisation of a graph homomorphism; the skew fibration property corresponds to maximality. Thus slicings are very closely related to combinatorial proofs.

References

[Abr07] S. Abramsky. *Interactive and Geometric Characterizations of the Space of Proofs (Abstract)*, volume 4646, pages 1–2. Springer, 2007.

[BHS05] R. F. Blute, M. Hamano, and P. J. Scott. Softness of hypercoherences and MALL full completeness. *Ann. Pure & Appl. Logic*, 131:1–63, 2005.

[CF05] Pierre-Louis Curien and Claudia Faggian. L-nets, strategies and proof-nets. In *Proc. CSL'05*, pages 167–183, 2005.

[CP05] J. Robin B. Cockett and Craig A. Pastro. A language for multiplicative-additive linear logic. *Elec. Notes in Theor. Comp. Sci.*, 122:23–65, 2005.

[CPS85] D.G. Corneil, Y. Perl, and L.K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Computing*, 14:926–934, 1985.

[CR79] S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Logic*, 44:36–50, 1979.

[Dan90] V. Danos. *La logique linéaire appliquée à l'étude de divers processus de normalisation et principalement du lambda calcul*. PhD thesis, Univ. de Paris, 1990.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir89] J.-Y. Girard. Towards a geometry of interaction. In *Categories in Computer Science and Logic*, volume 92 of *Contemporary Mathematics*, pages 69–108, 1989. Proc. of June 1987 meeting in Boulder, Colorado.

[Gir96] J.-Y. Girard. Proof-nets: the parallel syntax for proof theory. In *Logic and Algebra*, volume 180 of *Lecture Notes In Pure and Appl. Math.* Marcel Dekker, New York, 1996.

[Ham04] Masahiro Hamano. Softness of MALL proof-structures and a correctness criterion with mix. *Archive for Math. Logic*, 43:753–796, 2004.

[HG03] D. J. D. Hughes and R. J. van Glabbeek. Proof nets for unit-free multiplicative additive linear logic (Extended abstract). In *Proc. LICS'03*, pages 1–10. IEEE, 2003.

[HG05] D. J. D. Hughes and R. J. van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic. *ACM Transactions on Computational Logic (TOCL)*, 6:784–842, October 2005. Invited submission Nov. 2003, revised Jan. 2005, full version of [HG03].

[Hu99] H. Hu. Contractible coherence spaces and maximal maps. *Elec. Notes in Theor. Comp. Sci.*, 20, 1999.

[Hug06a] D. J. D. Hughes. Proofs without syntax. *Annals of Mathematics*, 143:1065–1076, 2006.

[Hug06b] D. J. D. Hughes. Towards Hilbert's 24^{th} Problem: Combinatorial Proof Invariants (Preliminary version). In *Proc. WOLLiC'06*, volume 165 of *Lec. Notes in Comp. Sci.*, 2006.

[Laf90] Y. Lafont. Interaction nets. In *Proc. 17-th ACM Symp. on Principles of Programming Languages, San Francisco*, pages 95–108, January 1990.

[LdF04] Olivier Laurent and Lorenzo Tortora de Falco. *Slicing polarized additive normalization*, volume 316, pages 247–282. LMS, 2004.

[Mai07] Roberto Maieli. Retractile proof nets of the purely multiplicative and additive fragment of linear logic. In *Proc. Logic Programming for AI and Reasoning*, volume 4790 of *LNAI*, pages 363–377. Springer-Verlag, 2007.

[New42] M. H. A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43:223–243, 1942.

[Sei74] S. Seinsche. On a property of the class of n-colorable graphs. *J. Combinatorial Th. (B)*, 16:191–193, 1974.

[Urq95] Alasdair Urquhart. The complexity of propositional proofs. *Bull. Symb. Logic*, 1:425–467, 1995.